



“LAB 11”

TANZEELA ASGHAR

2021-BSE-032 “A”

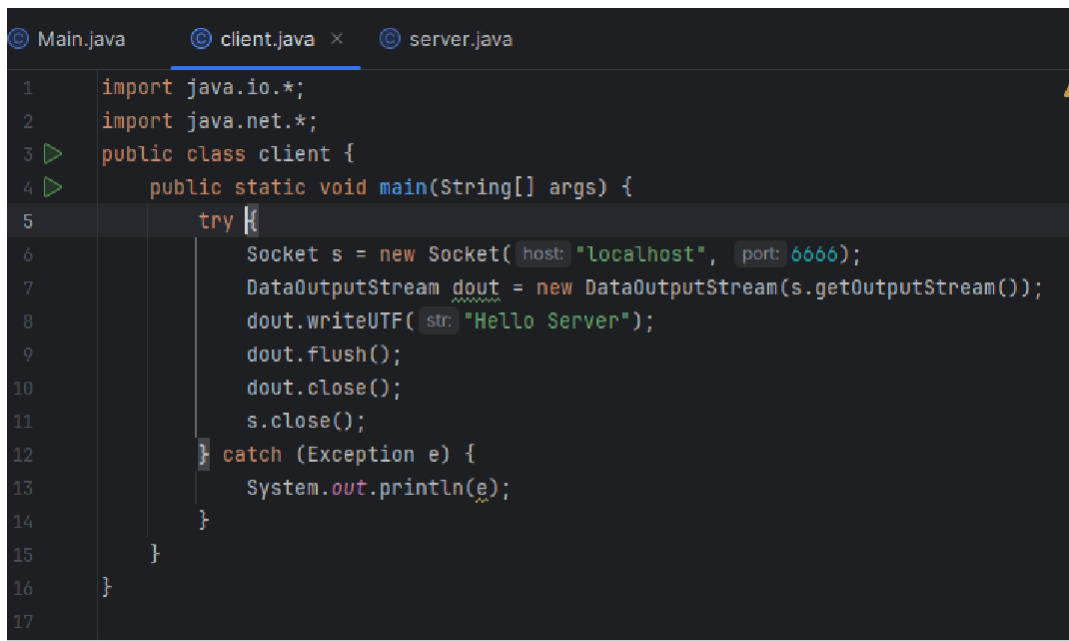
SUBMITTED TO: SIR SHAHZAD

EXAMPLE 1

SERVER

```
© Main.java    © client.java    © server.java ×
1
2  import java.io.*;
3  import java.net.*;
4  ▶ public class server {
5  ▶      public static void main(String[] args){
6          try{
7              ServerSocket ss=new ServerSocket( port: 6666);
8              Socket s=ss.accept();//establishes connection
9              DataInputStream dis=new DataInputStream(s.getInputStream());
10             String str=(String)dis.readUTF();
11             System.out.println("message= "+str);
12             ss.close();
13         }catch(Exception e){System.out.println(e);}
14     }
15 }
16
```

CLIENT

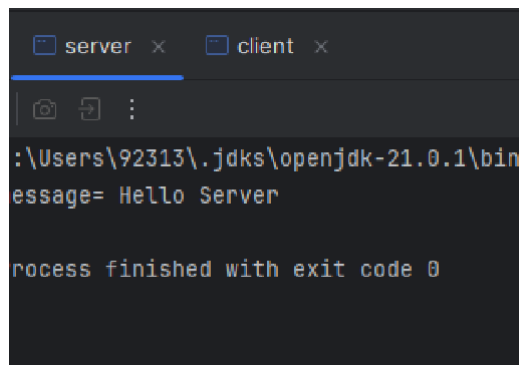


```

1  import java.io.*;
2  import java.net.*;
3  public class client {
4      public static void main(String[] args) {
5          try {
6              Socket s = new Socket(host: "localhost", port: 6666);
7              DataOutputStream dout = new DataOutputStream(s.getOutputStream());
8              dout.writeUTF(str: "Hello Server");
9              dout.flush();
10             dout.close();
11             s.close();
12         } catch (Exception e) {
13             System.out.println(e);
14         }
15     }
16 }
17

```

OUTPUT

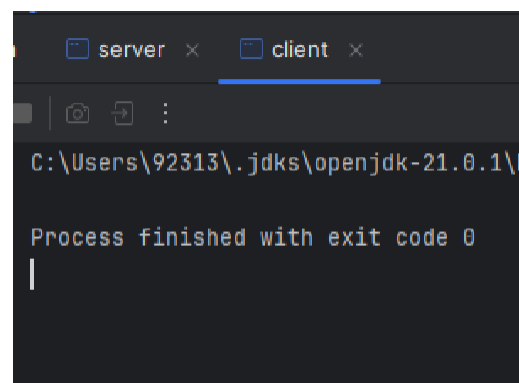


```

server x client x
: \Users\92313\jdk\openjdk-21.0.1\bin
message= Hello Server

Process finished with exit code 0

```



```

server x client x
C: \Users\92313\jdk\openjdk-21.0.1\

Process finished with exit code 0
|

```

EXAMPLE 2

SERVER

```

Main.java x client.java server.java x
1 import java.net.*;
2 import java.io.*;
3 class server{
4 public static void main(String args[])throws Exception{
5     ServerSocket ss=new ServerSocket( port: 3333);
6     Socket s=ss.accept();
7     DataInputStream din=new DataInputStream(s.getInputStream());
8     DataOutputStream dout=new DataOutputStream(s.getOutputStream());
9     BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
10
11     String str="",str2="";
12     while(!str.equals("stop")){
13         str=din.readUTF();
14         System.out.println("client says: "+str);
15         str2=br.readLine();
16         dout.writeUTF(str2);
17         dout.flush();
18     }
19     din.close();
20     s.close();
21     ss.close();
22 }

```

CLIENT

```

client.java x server.java
1 import java.net.*;
2 import java.io.*;
3 class client{
4 public static void main(String args[])throws Exception{
5
6     Socket s=new Socket( host: "localhost", port: 3333);
7     DataInputStream din=new DataInputStream(s.getInputStream());
8     DataOutputStream dout=new DataOutputStream(s.getOutputStream());
9     BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
10
11     String str="",str2="";
12     while(!str.equals("stop")){
13         str=br.readLine();
14         dout.writeUTF(str);
15         dout.flush();
16         str2=din.readUTF();
17         System.out.println("Server says: "+str2);
18     }
19
20     dout.close();
21     s.close();
22 }

```

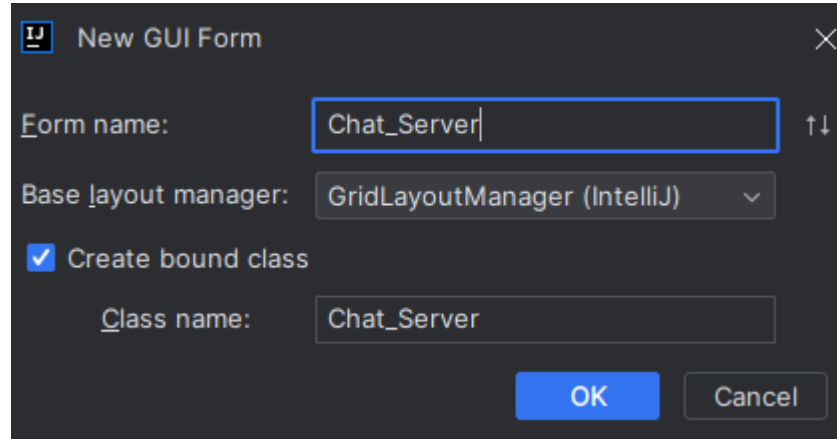
OUTPUT

```
C:\Users\92313\.jdk\openjdk-21.0.1\bin\java.exe
hello
Server says: hi
hiiiiiii
Server says:
|
```

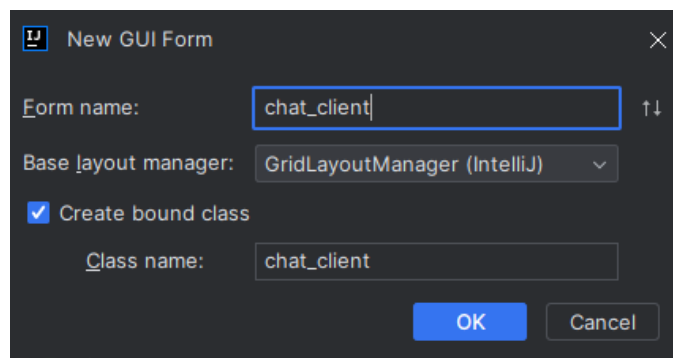
```
server x client x
:
:\Users\92313\.jdk\openjdk-21.0.1\bin\java.exe
lient says: hello
i
lient says: hiiiiiii
```

TASK:

BUILD CHAT APPLICATION USING SOCKET,SWING



The 'New GUI Form' dialog in IntelliJ is shown. The 'Form name' field contains 'Chat_Server'. The 'Base layout manager' is set to 'GridLayoutManager (IntelliJ)'. The 'Create bound class' checkbox is checked. The 'Class name' field also contains 'Chat_Server'. The 'OK' button is highlighted in blue.



The 'New GUI Form' dialog in IntelliJ is shown. The 'Form name' field contains 'chat_client'. The 'Base layout manager' is set to 'GridLayoutManager (IntelliJ)'. The 'Create bound class' checkbox is checked. The 'Class name' field also contains 'chat_client'. The 'OK' button is highlighted in blue.

CODE:

CLASS: CLIENT.JAVA

```
server.java  server.form  client.java x  client.form

1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import java.io.*;
6  import java.net.Socket;
7
8  public class client extends JFrame {
9      private JTextField textFieldclient;
10     private JTextArea clientmsg_area;
11     private JButton button1client;
12
13     private Socket socket;
14     private PrintWriter out;
15     private BufferedReader in;
```

```
server.java x  server.form  client.java x  client.form

17  public client() {
18      setTitle("Chat Client");
19      setSize( width: 400, height: 300);
20      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
21
22      textFieldclient = new JTextField();
23      clientmsg_area = new JTextArea();
24      button1client = new JButton( text: "Send");
25
26      setLayout(new BorderLayout());
27      add(textFieldclient, BorderLayout.SOUTH);
28      add(new JScrollPane(clientmsg_area), BorderLayout.CENTER);
29      add(button1client, BorderLayout.EAST);
30
31      button1client.addActionListener(new ActionListener() {
32          @Override
33          public void actionPerformed(ActionEvent e) {
34              sendMessage();
35          }
36      });
37
38      initializeClient();
39  }
```

```
server.java  server.form  client.java x  client.form

40
1 usage
41 private void initializeClient() {
42     try {
43         socket = new Socket( host: "localhost", port: 12345);
44         out = new PrintWriter(socket.getOutputStream(), autoFlush: true);
45         in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
46
47         new Thread(new ReceiveMessage()).start();
48     } catch (IOException e) {
49         e.printStackTrace();
50     }
51 }
52
1 usage
53 private void sendMessage() {
54     String message = textFieldclient.getText();
55     if (!message.isEmpty()) {
56         out.println("Client: " + message);
57         textFieldclient.setText("");
58     }
59 }
60
```

```
server.java  server.form  client.java x  client.form

1 usage
61 private class ReceiveMessage implements Runnable {
62     @Override
63     public void run() {
64         try {
65             String receivedMessage;
66             while ((receivedMessage = in.readLine()) != null) {
67                 clientmsg_area.append(receivedMessage + "\n");
68             }
69         } catch (IOException e) {
70             e.printStackTrace();
71         }
72     }
73 }
74
75 public static void main(String[] args) {
76     SwingUtilities.invokeLater(new Runnable() {
77         @Override
78         public void run() {
79             new client().setVisible(true);
80         }
81     });
82 }
83
84
```

CLASS: SERVER.JAVA

```
server.java x server.form client.java client.form
1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5 import java.io.*;
6 import java.net.ServerSocket;
7 import java.net.Socket;
8
9 public class server extends JFrame {
10     private JTextArea msg_area;
11     private JTextField textFieldserver;
12     private JButton buttonserver;
13
14     private ServerSocket serverSocket;
15     private Socket clientSocket;
16     private PrintWriter out;
17     private BufferedReader in;
```

```
server.java x server.form client.java client.form
19 public server() {
20     setTitle("Chat Server");
21     setSize( width: 400, height: 300);
22     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23
24     msg_area = new JTextArea();
25     textFieldserver = new JTextField();
26     buttonserver = new JButton( text: "Send");
27
28     setLayout(new BorderLayout());
29     add(new JScrollPane(msg_area), BorderLayout.CENTER);
30     add(textFieldserver, BorderLayout.SOUTH);
31     add(buttonserver, BorderLayout.EAST);
32
33     buttonserver.addActionListener(new ActionListener() {
34         @Override
35         public void actionPerformed(ActionEvent e) {
36             sendMessage();
37         }
38     });
39
40     initializeServer();
41 }
```



```
server.java x server.form client.java client.form
42
1 usage
43 private void initializeServer() {
44     try {
45         serverSocket = new ServerSocket(port: 12345);
46         msg_area.append("Server started. Waiting for a client...\n");
47
48         clientSocket = serverSocket.accept();
49         msg_area.append("Client connected.\n");
50
51         out = new PrintWriter(clientSocket.getOutputStream(), autoFlush: true);
52         in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
53
54         new Thread(new ReceiveMessage()).start();
55     } catch (IOException e) {
56         e.printStackTrace();
57     }
58 }
```

```
server.java x server.form client.java client.form
60 private void sendMessage() {
61     String message = textFieldserver.getText();
62     if (!message.isEmpty()) {
63         msg_area.append("Server: " + message + "\n");
64         out.println("Server: " + message);
65         textFieldserver.setText("");
66     }
67 }
68
1 usage
69 private class ReceiveMessage implements Runnable {
70     @Override
71     public void run() {
72         try {
73             String receivedMessage;
74             while ((receivedMessage = in.readLine()) != null) {
75                 msg_area.append(receivedMessage + "\n");
76             }
77         } catch (IOException e) {
78             e.printStackTrace();
79         }
80     }
81 }
```

```

82
83  ▶ public static void main(String[] args) {
84      SwingUtilities.invokeLater(new Runnable() {
85          @Override
86  ↗ public void run() {
87          new server().setVisible(true);
88      }
89      });
90  }
91  }

```

OUTPUT:

